**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Основы программирования»

Отчет по лабораторной работе № 10

«Вычисление обратной матрицы методом Гаусса-Жордана»

| Выполнил: Чернев Николай Андреевич |  | Проверил: |
| --- | --- | --- |
| студент группы ИУ5-14Б |  | преподаватель каф. ИУ5 |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2023 г.

**Задание**

Создать функцию для вычисления обратной матрицы по методу Гаусса-Жордана. Размер матрицы передавать в функцию в качестве параметра. Для упрощения алгоритма следует присоединить единичную матрицу справа к исходной и выполнять все преобразования над объединенной матрицей размером N\*2N. Обратная матрица получится на месте единичной в столбцах N…2N, а на месте исходной матрицы в столбцах 0…(N-1) должна получиться единичная матрица.

Включить в алгоритм проверку на существование обратной матрицы. Для этого в в прямом ходе перед делением выполнить проверку на ноль элементов главной диагонали исходной матрицы. Если элемент равен 0, то нужно поменять местами текущую строку с одной из нижележащих строк, в которой элемент в соответствующем столбце не равен 0.

Если таких строк нет, то выдать сообщение: «Обратная матрица не существует».

Применить функцию для решения системы линейных алгебраических уравнений.

**Разработка алгоритма**

Описание переменных

Функция search\_rev\_mtx:

* bool is\_rev\_exist - отвечает за наличие обратной матрицы(true - есть обратная матрица, false - нет)
* int i, j - счетчики циклов

Функция slay\_solving:

* double \*ans - массив корней СЛАУ
* int i - счетчик циклов

Функция main:

* int n - размер матрицы
* double \*\*mtx - матрица, обратную к которой надо найти
* int i, j - счетчики циклов

**Описание функций**

void div\_row(double \*row, int n, double d) - принимает row - строку матрицы, n - размер матрицы, d - делитель; делит каждое число в строке на d, ничего не возвращает

void sum\_rows(double \*row, const double \*source, int n, double k) - принимает 2 строки матрицы - row и source, n - размер матрицы и множитель k; прибавляет к каждому элементу из row соответствующий элемент из source, умноженный на k, ничего не возвращает

bool search\_rev\_mtx(double \*\*mtx, int n) - принимает матрицу mtx и ее размер n; вычисляет обратную матрицу к данной методом Гаусса-Жордана, возвращает true - если у данной матрицы есть обратная, false - если у данной матрицы нет обратной

void slay\_solving(double \*\*mtx, double \*free\_coeffs, int n) - принимает матрицу коэффициентов mtx, массив свободных членов СЛАУ free\_coeffs и размер матрицы n; находит корни СЛАУ и выводит их, ничего не возвращает

**Код программы**

#include <iostream>

#include <iomanip>

using namespace std;

void div\_row(double \*row, int n, double d) {

for (int i = 0; i < 2 \* n; i++) {

row[i] /= d;

}

}

void sum\_rows(double \*row, const double \*source, int n, double k) {

for (int i = 0; i < 2 \* n; i++) {

row[i] += source[i] \* k;

}

}

bool search\_rev\_mtx(double \*\*mtx, int n) {

bool is\_rev\_exist;

for (int i = 0; i < n; i++) {

if (mtx[i][i] == 0) {

is\_rev\_exist = false;

for (int j = i + 1; (j < n) && (!is\_rev\_exist); j++) {

if (mtx[j][i] != 0) {

swap(mtx[i], mtx[j]);

is\_rev\_exist = true;

}

}

if (!is\_rev\_exist) {

return false;

}

}

else if (mtx[i][i] != 1) {

div\_row(mtx[i], n, mtx[i][i]);

}

for (int j = 0; j < n; j++) {

if (j != i) {

sum\_rows(mtx[j], mtx[i], n, -mtx[j][i]);

}

}

}

return true;

}

void slay\_solving(double \*\*mtx, double \*free\_coeffs, int n) {

double \*ans = new double[n];

for (int i = 0; i < n; i++) {

ans[i] = 0;

for (int j = 0; j < n; j++) {

ans[i] += mtx[i][j + n] \* free\_coeffs[j];

}

}

for (int i = 0; i < n; i++) {

cout << 'x' << i + 1 << " = " << ans[i] << '\n';

}

delete[] ans;

}

int main() {

int n;

cout << "Введите n - размер матрицы\n";

cin >> n;

double \*\*mtx = new double \*[n];

for (int i = 0; i < n; i++) {

mtx[i] = new double[2 \* n];

}

cout << "Введите матрицу\n";

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cin >> mtx[i][j];

mtx[i][j + n] = (i == j) ? 1 : 0;

}

}

if (search\_rev\_mtx(mtx, n)) {

cout << "Обратная матрица:\n";

for (int i = 0; i < n; i++) {

for (int j = n; j < 2 \* n; j++) {

cout << setw(8) << setprecision(5) << mtx[i][j] << ' ';

}

cout << '\n';

}

cout << '\n';

double \*free\_coeffs = new double[n];

cout << "Введите свободные члены СЛАУ\n";

for (int i = 0; i < n; i++) {

cin >> free\_coeffs[i];

}

slay\_solving(mtx, free\_coeffs, n);

delete[] free\_coeffs;

}

else {

cout << "Обратной матрицы не существует\n";

}

for (int i = 0; i < n; i++) {

delete[] mtx[i];

}

delete[] mtx;

return 0;

}

**Анализ результатов**